## Motivation

- Data is often stored at untrusted servers
  - Data in the cloud
  - Insecure server
- Can we establish the trustworthiness of data from these servers? i.e.:
  - Authenticity of retrievals
  - Integrity of data (updates)
  - Secure provenance of data
  - Indemnity for the server (cloud)

## Model

**Untrusted Server (Bob)**
- Data
- Authentication DS

**Trusted Server (Alice)**
- Client (Carol)
- Client (Carol)
- Client (Carol)

## Protocol for Static Data

**Data is static**

- Only Alice can modify data

**Alice**
- Data
- Compute P
- \( P = P' \)
- Declare P

**Bob**
- Data
- AuthDS
- Query
- Results
- Verify
- Process Query
- Update AuthDS and store previous values
- Compute data read by the transaction
- Compare

**Carol**
- Data
- Compute P
- \( P = P' \)
- Declare P
- Proof P

## Challenge: Dynamic Data

Clients can modify data. No centralized vetting of updates

- A trusted server is used to keep track of proofs

**Alice**
- Data
- Computer P
- \( P = P' \)
- Public P List
- Compare

**Bob**
- Data
- AuthDS
- Process Query
- New P
- Update AuthDS and store previous values
- Compute data read by the transaction
- Compare

**Carol1**
- Query
- Results
- New Proof
- Verify

**Carol2**
- Verification Object
- Compute P
- Compare

## Experiments

Easy to implement on top of an existing DBMS (e.g. Oracle)

- MB1 with 1 client
- MB1 with 5 clients
- MB1 with 1 client
- MB1 with 5 clients

**Fig 3: Legends**

**Fig 1: Insert + Verification Time**

**Fig 2: Insert Time**

**Fig 3: Search + Verification Time**

**Fig 4: Search + Verification Time**

## Conclusion

- Protocols provide authenticity, integrity and indemnity for relational databases
- Significantly reduces level of trust required
- Provides secure provenance of data
- Verification is decoupled from transaction execution
- Easy to implement and reasonable overhead